

- हेडिंग: "Bank Account Form"

html

CopyEdit

```
<select id="accountType">
  <option value="savings">Savings Account</option>
  <option value="current">Current Account</option>
</select>
```

- ड्रॉपडाउन मेनू: उपयोगकर्ता को **Savings या Current Account** चुनने का विकल्प देता है।

html

CopyEdit

```
<input type="text" id="name" placeholder="Owner Name" />
<input type="number" id="initial" placeholder="Initial Balance" />
<button onclick="createAccount()">Create Account</button>
```

- इनपुट फ़ील्ड:
 - नाम डालने के लिए।
 - प्रारंभिक बैलेंस डालने के लिए।
 - "Create Account" बटन क्लिक करने पर `createAccount()` फ़ंक्शन चलता है।

html

CopyEdit

```
<hr>
```

- एक क्षैतिज लाइन जो दो सेक्शन को अलग करती है।

html

CopyEdit

```
<input type="number" id="amount" placeholder="Enter Amount" />
<button onclick="deposit()">Deposit</button>
<button onclick="withdraw()">Withdraw</button>
<button onclick="applyInterest()">Apply Interest (Savings
only)</button>
```

- अमाउंट डालने के लिए इनपुट बॉक्स और तीन बटन:
 - `deposit()` = जमा करने के लिए।
 - `withdraw()` = निकालने के लिए।
 - `applyInterest()` = ब्याज जोड़ने के लिए (सिर्फ Savings के लिए)।

html

CopyEdit

```
<div class="output" id="output">No account yet.</div>
```

- इस डिव में आउटपुट मैसेज और अकाउंट की जानकारी दिखाई जाती है।

◆ JavaScript सेक्शन (लॉजिक)

javascript

CopyEdit

```
class BankAccount {
  #accountNumber;
  #owner;
  #balance;
```

- `BankAccount` क्लास में **private fields** (सिर्फ अंदर से एक्सेसिबल) बनाए गए हैं: खाता नंबर, मालिक का नाम, और बैलेंस।

javascript

CopyEdit

```
  constructor(accountNumber, owner, initialBalance = 0) {
    if (new.target === BankAccount) throw new Error("Abstract
class!");
```

- यह एब्सट्रैक्ट क्लास है — सीधे इस्तेमाल नहीं किया जा सकता। सबक्लास से ही काम करेगा।

javascript

CopyEdit

```
    this.#accountNumber = accountNumber;
    this.#owner = owner;
```

```
    this.#balance = initialBalance;
}
```

- कंस्ट्रक्टर इनिशियल बैल्यूज़ सेट करता है।

javascript

CopyEdit

```
getBalance() { return this.#balance; }
getInfo() { return `Owner: ${this.#owner} | Balance:
₹${this.#balance}`; }
_changeBalance(amount) { this.#balance += amount; }
```

- `getBalance()` = बैलेंस दिखाता है।
 - `getInfo()` = मालिक और बैलेंस की स्ट्रिंग लौटाता है।
 - `_changeBalance()` = बैलेंस बढ़ाने या घटाने का helper method है।
-

◆ SavingsAccount क्लास

javascript

CopyEdit

```
class SavingsAccount extends BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    super(accountNumber, owner, balance);
    this.interestRate = 0.03;
  }
}
```

- बचत खाता क्लास, जिसमें 3% ब्याज सेट किया गया है।

javascript

CopyEdit

```
deposit(amount) {
  this._changeBalance(amount);
  return `Deposited ₹${amount}`;
}
```

- जमा किया गया अमाउंट बैलेंस में जोड़ा जाता है और मैसेज रिटर्न होता है।

javascript

CopyEdit

```
withdraw(amount) {  
  if (amount <= this.getBalance()) {  
    this._changeBalance(-amount);  
    return `Withdraw ₹${amount}`;  
  } else {  
    return "Insufficient funds.";  
  }  
}
```

- बैलेंस चेक करके अमाउंट निकाला जाता है।

javascript

CopyEdit

```
applyInterest() {  
  const interest = this.getBalance() * this.interestRate;  
  this._changeBalance(interest);  
  return `Interest ₹${interest.toFixed(2)} applied.`;  
}  
}
```

- ब्याज जोड़ता है और मैसेज देता है।

◆ CurrentAccount क्लास

javascript

CopyEdit

```
class CurrentAccount extends BankAccount {  
  constructor(accountNumber, owner, balance = 0) {  
    super(accountNumber, owner, balance);  
    this.overdraftLimit = 10000;  
  }  
}
```

- चालू खाता जिसमें ₹10,000 की ओवरड्राफ्ट लिमिट होती है।

javascript

CopyEdit

```
deposit(amount) {
  this._changeBalance(amount);
  return `Deposited ₹${amount}`;
}

withdraw(amount) {
  if (amount <= this.getBalance() + this.overdraftLimit) {
    this._changeBalance(-amount);
    return `Withdrew ₹${amount}`;
  } else {
    return "Overdraft limit exceeded.";
  }
}
```

- जमा: सीधा बैलेंस बढ़ा देता है।
- निकासी: बैलेंस + लिमिट के भीतर है तो निकाल सकता है, वरना एरर देता है।

◆ App Logic (Functions for Buttons)

javascript

CopyEdit

```
let account = null;
let accId = 100;
```

- एक ग्लोबल वेरिएबल `account` है जो एक्टिव खाता को स्टोर करता है।
- `accId` से हर नए खाते को यूनिक ID दी जाती है।

javascript

CopyEdit

```
function createAccount() {
  const type = document.getElementById('accountType').value;
  const name = document.getElementById('name').value;
  const initial =
parseFloat(document.getElementById('initial').value) || 0;
```

```

if (type === 'savings') {
    account = new SavingsAccount("S" + (++accId), name, initial);
} else {
    account = new CurrentAccount("C" + (++accId), name, initial);
}

show(`Account created for ${name} with ₹${initial}`);
}

```

- ड्रॉपडाउन से अकाउंट टाइप, नाम और प्रारंभिक बैलेंस लेता है।
- फिर Savings या Current क्लास से अकाउंट बनाता है।

javascript

CopyEdit

```

function deposit() {
    const amt = parseFloat(document.getElementById('amount').value);
    if (account && amt > 0) {
        show(account.deposit(amt));
    } else {
        show("Invalid deposit.");
    }
}

```

- अमाउंट लेकर जमा करता है।

javascript

CopyEdit

```

function withdraw() {
    const amt = parseFloat(document.getElementById('amount').value);
    if (account && amt > 0) {
        show(account.withdraw(amt));
    } else {
        show("Invalid withdrawal.");
    }
}

```

- अमाउंट लेकर निकासी करता है।

javascript

CopyEdit

```
function applyInterest() {  
  if (account instanceof SavingsAccount) {  
    show(account.applyInterest());  
  } else {  
    show("Interest can only be applied to Savings Account.");  
  }  
}
```

- सिर्फ Savings अकाउंट में ब्याज लगाता है।

javascript

CopyEdit

```
function show(message) {  
  const output = document.getElementById('output');  
  output.innerHTML = `${message}<br>${account ? account.getInfo() : ''}`;  
}
```

- आउटपुट सेक्शन में मैसेज और अकाउंट की जानकारी दिखाता है।

form ek **simple banking simulator** hai jisme:

1. **User** select karta hai:

- Account Type: **Savings** ya **Current**
- Name & Initial Balance

2. Account create hota hai **SavingsAccount** ya **CurrentAccount** class se

3. Uske baad:

- **Deposit:** User paisa daal sakta hai
- **Withdraw:** Paisa nikaal sakta hai
- **Apply Interest:** Sirf savings account ke liye (3% extra balance)

- Sab kuch form ke through hota hai, live output ke saath

Isme Encapsulation Ka Role:

Encapsulation ka matlab hota hai:

"Data ko private banana aur access sirf methods ke through dena."


Code Example:

```
js
CopyEdit
#balance; // private field

getBalance() {
  return this.#balance;
}
```

Benefits:

- `#balance` ko direct modify **koi bhi nahi** kar sakta (form ya developer bhi nahi)
- Sirf `deposit()`, `withdraw()`, `getBalance()` ke through access possible hai
- Data ka **unauthorized access ya manipulation** nahi hota

 Yani internal data safe hai — ye hai **Encapsulation**

Isme Abstraction Ka Role:

Abstraction ka matlab hota hai:

"User ko sirf zaroori features dikhana, internal logic chhupana."

Code Example:


```
js
CopyEdit
class BankAccount {
  deposit() {
```



```
        throw new Error("Must implement in subclass");
    }
}
```

Benefits:

- `BankAccount` class ek abstract base hai — **direct use nahi hoti**
- Sirf subclasses (`SavingsAccount`, `CurrentAccount`) real accounts hain
- User ko `applyInterest()` ka button diya gaya — lekin uske peeche ka logic chhupa hua hai
- Har account apne hisaab se deposit/withdraw handle karta hai

 Yani user ko **clean interface milta hai**, internal complexity dikhayi nahi deti — ye hai **Abstraction**

Final Summary Table:

Feature	Code / Form Me Kaha Hai
Encapsulation	<code>#balance</code> , <code>#owner</code> jaise private fields + <code>getBalance()</code>
Abstraction	<code>BankAccount</code> base class (abstract) + <code>deposit()/withdraw()</code> exposed only
Form Function	UI se account create, deposit, withdraw, interest apply

Bhai tu is form ko apne project me bhi use kar sakta hai, aur agar tu chahe toh mai isme:

- Transaction history
- Styled design (Tailwind / Bootstrap)
- Multi-user support

bhi add karwa sakta hoon. Bas bol de kya chahiye